Cricket umpire assistance and ball tracking system using a single smartphone camera

Udit Arora¹, Sohit Verma¹, Sarthak Sahni¹, Tushar Sharma¹
¹Division of Computer Engineering, Netaji Subhas Institute of Technology, New Delhi 110078, India

Corresponding Author:
Udit Arora¹
Email address: udita1.co@nsit.net.in

ABSTRACT
Several ball tracking algorithms have been reported in literature. However, most of them use high-quality video and multiple cameras, and the emphasis has been on coordinating the cameras or visualizing the tracking results. This paper aims to develop a system for assisting the umpire in the sport of Cricket in making decisions like detection of no-balls, wide-balls, leg before wicket and bouncers, with the help of a single smartphone camera. It involves the implementation of Computer Vision algorithms for object detection and motion tracking, as well as the integration of machine learning algorithms to optimize the results.

Techniques like Histogram of Gradients (HOG) and Support Vector Machine (SVM) are used for object classification and recognition. Frame subtraction, minimum enclosing circle, and contour detection algorithms are optimized and used for the detection of a cricket ball. These algorithms are applied using the Open Source Python Library - OpenCV. Machine Learning techniques - Linear and Quadratic Regression are used to track and predict the motion of the ball. It also involves the use of open source Python library VPython for the visual representation of the results. The paper describes the design and structure for the approach undertaken in the system for analyzing and visualizing off-air low-quality cricket videos.

1. INTRODUCTION
In cricket, an umpire is a person who has the authority to make judgments on the cricket field, according to the laws of cricket. The umpire's decision depends on many criteria, including where the ball pitched, whether the ball hit in line with the wickets and whether the batsman was attempting to hit the ball.

The game of cricket is widely played and followed in India and many other countries in the world. There are millions of amateur and aspiring cricketers involved. But the high cost and amount of technological requirements of tracking technologies restrict their use in any matches, competitions and training academies other than the ones operating at an international level. Computer vision seems like a natural choice for these applications. In spite of the success of computer vision technology in several other fields, there are very few computer vision systems and algorithms which enhance the experience of cricket which operate at a low cost.
A promising research direction is the use of computer vision to detect, identify and track the cricket ball (and other relevant objects in the context of cricket), and machine learning techniques to optimize and further predict various results and decisions. The use of just one camera, which may be of a quality equivalent to modern day smartphone cameras, along with the various algorithms and techniques of Computer Vision and Machine Learning can help us achieve a system that reliably assists the umpire and operates at a cheap cost.

This paper aims to develop a low-cost computer system which assists the umpire in cricket, operates at a low cost, has lesser technological (software and hardware) requirements, and can be used at sub-international levels in the sport of cricket.

2. RELATED WORK
In the past few decades, researchers and engineers have developed quite a few technologies, one of the prominent ones being Hawkeye, as explored by Bal and Dureja [1], for assisting the umpire in making decisions. These solutions have mostly involved the use of multiple high resolution and high frame rate cameras. In most cases, they have been developed keeping in mind the professional and international level of cricket.

Although many object tracking techniques have been reported in literature, very few focus on cricket ball tracking for umpiring assistance. Khan, Aalsalem, and Arshed [2] proposed an aware cricket ground system that focused on finding the location of the ball during the game at any given instance and thus making the desired calculations whenever needed for analysis. Yan et al. [3] designed an automatic tennis annotation system that tracked the tennis ball at high speeds, along with the incorporation of detection of key events and player positions, to generate annotations using a hidden Markov model applied to the events.

3. APPROACH
Figure 1 below shows a flowchart of our umpire assistance system. It consists of multiple stages. As a pre-processing stage, image frames are extracted from the video, and the ball is detected in each of the frames. The ball tracking module keeps track of the ball position following which the 3D mapping module maps the tracked ball position from 2D into 3D. This 3D data is then smoothened out using regression techniques. Finally, the visualization module uses this mapped 3D data to determine the umpiring decisions and visualize the results in 3D.
3.1 Ball Detection

**Proposed Algorithm**

The cricket ball is detected using Support Vector Machine (SVM) and Histogram of Oriented Gradient (HOG). Positive and negative data samples are collected and used for building the
SVM models for HOG objects (ball, batsman). The video is sliced into multiple image frames separated by a fixed time duration. Frame difference technique is used to find areas which may have a high possibility of having a ball. Each window frame, after conversion to grayscale, is used to extract HOG features and fed to the SVM model to detect the ball in the areas found to be different in subsequent frames by the frame subtraction method.

Training Data Preparation

Data samples are collected by manually scraping through the captured videos and taking screenshots of relevant objects. There are two types of samples: negative and positive. Negative samples correspond to non-object images. Positive samples correspond to images with detected objects.

Negative Samples

Negative samples are taken from arbitrary images. After taking a few negative samples, hard negative mining is applied, i.e., manually including all the false positive detections made by the SVM into the negative data sample set. A user interface was developed for the same which made the task easier and more efficient.

Positive Samples

Positive samples are collected by taking screenshots of a standardized size of the cricket ball from the captured video data. The positive sample set consists of a large number of images of a cricket ball differing in various parameters like orientation, color intensity, brightness, background, etc. The object instances are taken as screenshot images from the videos. Then they are resized to target samples size and stored in the positive sample set. No distortion is applied.

Histogram of Oriented Gradients (HOG)

OpenCV provides the functionality of extracting the HOG features of images using the various parameters mentioned below.

- Minimum Window Size (min_wdw_sz) signifies the size of the window of which the HOG features are to be extracted.
- Step Size (step_size) indicates the amount by which the window moves in one iteration.
- Pixels Per Cell (pixels_per_cell) is the number of pixels that are there in a cell.
- Cells per block (cells_per_block) is the number of cells in one block of the window.
Building the SVM

HOG features are extracted from the collected positive and negative data samples. The SVM model is built using the OpenCV SVM library using the extracted HOG features. As a result, we get an SVM model which is used to classify objects into those that are a ball and those that are not.

3.2 Ball Tracking

Sliding Window Approach

The window where the ball is detected in a particular frame is used in the subsequent frame as a reference, and the SVM model is used to detect the presence of the ball in an inflated window in the neighborhood of the previous frame’s window. This approach ensures higher accuracy in ball detection while improving performance.
The obtained position of the ball is stored as a set of ‘x’ and ‘y’ coordinates in a separate file.

**Bounce Point Detection**

The point where the ball bounces is where the ‘y’ coordinate is the least.

**Rejection of False Detections**

Consecutive detections separated by a value less than a maximum distance are included in our correct detections. If the distance is greater than a particular value, the difference of angle between following points on the trajectory path followed by the ball is used, and false detections are rejected using a maximum permissible value of this angle. Further, some points are rejected while mapping to 3D coordinates based on the threshold values and the \((x, y, \text{radius})\) of the detected ball.
Detection of Ball Hitting the Bat or Batsman

The above-mentioned maximum distance and angle difference methods are used to detect drastic changes in path due to the impact of the bat or batsman’s body on the ball. All detections made after that would be false and hence are rejected.

3.3 Batsman Detection and Tracking

The batsman is detected using an approach similar to that used for ball detection. OpenCV’s built-in People Detector SVM models are used. Further, NMS is applied to the detected objects to obtain the final result. The movement of the batsman is tracked using a similar approach, i.e., limiting the search window to the neighborhood of the first detection. The detected batsman window is displayed on the frame.

Fig. 9: The tracked batsman after NMS

Fig. 10: Batsman tracking logs across frames
3.4 Mapping 2D Image coordinates to 3D world coordinates

Fig. 11: Finding x, y, and radius of the tracked ball

1. The x, y coordinates are obtained from the tracked ball using the **Minimum Enclosing Circle** algorithm after a series of image transformations.
   a. The tracked ball window is cropped and converted to grayscale.

Fig. 12: The tracked ball in color and converted to grayscale

b. CLAHE is used to improve the contrast of the image, and the image is blurred using Gaussian Blur technique.

Fig. 13: Result of CLAHE algorithm and Gaussian blurring

c. The average color around the center of the frame is found and used to find a suitable threshold brightness value to extract the ball from the frame. The frame is
then thresholded by setting the intensity of the pixels with value below the
threshold to 1, and those with a value above the threshold to 0.

![Tracked Ball](image1.png)

Fig. 14: Result of thresholding

d. Contour detection is now used to find the boundary of the tracked ball and the
maximum sized contour closest to the center of the frame is selected.

![Contours](image2.png)

Fig. 15: Detected Contours

e. Minimum enclosing circle algorithm is now applied on the detected contour to
find the center and radius of the tracked ball.

![Best Fit Circle](image3.png)

Fig. 16: Minimum enclosing circle shown on blurred image

2. The obtained (x, y) image coordinates are scaled to world coordinates according to the
cricket pitch dimensions.

3. The z-coordinate (depth) is obtained by comparing the radius of the tracked ball with the
radius obtained at the beginning of the pitch vs. the radius at the end of the pitch and
scaling this ratio to the pitch length.

4. The scaled (x, y) coordinates are transformed using perspective projection to obtain the
real world (x, y) coordinates.

5. Weighted linear and quadratic regression is applied across various dimensions to
smoothen out the obtained coordinates for better visualization.

6. The trajectory of the ball is predicted further using the polynomial equations obtained via
regression and the predicted ball coordinates are found.
3.5 Umpiring Decisions

Information related to the ball and batsman is calculated using their obtained coordinates.

- The speed of the ball is found by taking an average of the speed of the ball before the bouncing point (if available) or after.
- The height of the batsman is found by finding the midpoint of the detected batsman window, mapping the midpoint to real-world coordinates, and taking a double of the real-world height.

The obtained coordinates and other information is then used for umpiring decisions.

3.5.1 Leg Before Wicket (LBW) Decision

1. The impact location of the ball with the batsman is checked to see if it is inside the impact zone.
2. The pitching location of the ball is found.
3. The location of the ball when it is nearest to the wickets is found from the predicted ball trajectory, and this location is then checked to see if the ball would hit the wickets.

These three conditions are checked to arrive at the final LBW decision.

3.5.2 Wide Decision

1. The location of the ball when it is nearest to the batsman’s wicket crease is found.
2. If the location is towards the leg side of the batsman, it is marked as Wide.
3. If the location is towards the off side of the batsman, the location is compared with the location of the Wide crease line.
4. If the height of the ball near the crease is greater than the height of the batsman, it is marked as wide.

3.5.3 No Ball Decision

1. The location of the ball when it is nearest to the batsman’s wicket crease is found.
2. If the ball hasn’t bounced off the ground, and the height of the ball is greater than half of the batsman’s height, it is marked as a no-ball.

3.5.4 Bouncer Decision

1. The location of the ball when it is nearest to the batsman’s wicket crease is found.
2. If the ball has a valid bounce point, and the height of the ball is greater than shoulder height of the batsman, it is marked as a bouncer.

3.6 3D Visualization of Tracked Objects and Cricket Pitch

Using the object (x, y, z) coordinates obtained by mapping algorithm, the objects are visualized in a 3D space using VPython visualization library.

Visualization

1. The pitch and the wickets are drawn by using boxes, taking ideal pitch measurements.
2. The tracked ball is displayed on the screen using spheres with a cylindrical trajectory.
3. The trajectory of the ball is predicted further using the polynomial equations obtained via regression and the predicted ball positions are displayed on the screen using spheres of a different color than the detected balls.
4. The various decision parameters are shown on the screen.

Fig. 18: The visualization result
5. FFMPEG was used to create a video of the visualization in action, which can be seen here - https://youtu.be/K9MP5U1GBEI

3.7. Graphical User Interface for the Application

A GUI for the system was developed using the Kivy python framework.

![Image: GUI on first load]

**Implementation**

- The **Videos** section allows selection of a video to analyze and visualize.
- The selected video’s full name is displayed under the heading ‘**Selected Video**’
- The **Select Bowling Action** dropdown button is used to choose the action between fast, slow and no-action for debugging purposes.
- The **Sliding Windows** toggle button when toggled shows the sliding windows in progress during ball detection.
- The **Play Video** button plays the selected video using a video player.
- The **Analyze** button activates the script that analyzes the video and finds the image coordinates of the tracked ball and player across frames.
- The **Visualize** button activates the script that visualizes the results of the analysis in 3D.
4. EXPERIMENTS

The videos used for developing and testing the system were taken from practice sessions of the NSIT cricket team. Although a wide variety of videos were captured, 100+ in total, the size of the data wasn’t enough for us to statistically determine the accuracy of the system. The system was later even deployed and used by the college team’s coach during their practice sessions to improve player performance.

5. CONCLUSION

The primary objective of this paper was to develop a product for assisting the umpire in the sport of cricket while making decisions, using a single camera. It involved the development of algorithms using computer vision and machine learning techniques for ball detection and tracking, along with various cricket decision-making rules.

The paper discusses the use of computer vision to detect, identify and track the cricket ball (and other relevant objects in the context of cricket), and machine learning techniques to optimize and further predict various results and decisions. The use of just one camera, which may be of a quality equivalent to modern day smartphone cameras, along with the various algorithms and techniques of Computer Vision and Machine Learning helped us achieve a system that reliably assists the umpire and operates at a cheap cost.

The feature Histogram of Oriented Gradients (HOG) is implemented along with a Support Vector Machine (SVM) model to classify and detect the ball and batsman in a window
frame. OpenCV and its various image processing features, with algorithmic techniques like frame subtraction, sliding windows, CLAHE, minimum enclosing circle and machine learning techniques of weighted regression are implemented to achieve accurate tracking of the ball in motion.

Various umpiring decisions are made by checking specific conditions on the data obtained as per the rules of the sport of cricket and the decisions involved. VPython module is used to represent the decisions in the form of interactive 3D visualizations showing the ball trajectory path and information about the umpiring decisions. Kivy is used to provide an interactive graphical user interface to access and use the tool.

REFERENCES

